Department of Computer Science & Engineering
BRAC University.

Interrupts

Course ID: CSE341
Course Title: Microprocessors

Lecture References:

- Book:
  - *Microprocessors and Interfacing: Programming and Hardware,* **Author:** Douglas V. Hall
  - *Microprocessor and Microcomputer – Based System Design*, **Author:** Mohamed Rafiquzzaman

Interrupt

- Interrupt is a process where a normal program execution to be interrupted by some external signal or by a special instruction in the program.

- Microprocessor pay attention to the interrupt stopping the current execution.

- **What happens when MP is interrupted ?**
  - When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an **Interrupt Service Routine** (ISR) to respond to the incoming interrupt.
  - Each interrupt will have its own ISR.
  - After finishing the second program/interrupt, automatically return to the first program and start execution from where it was left

Interrupt

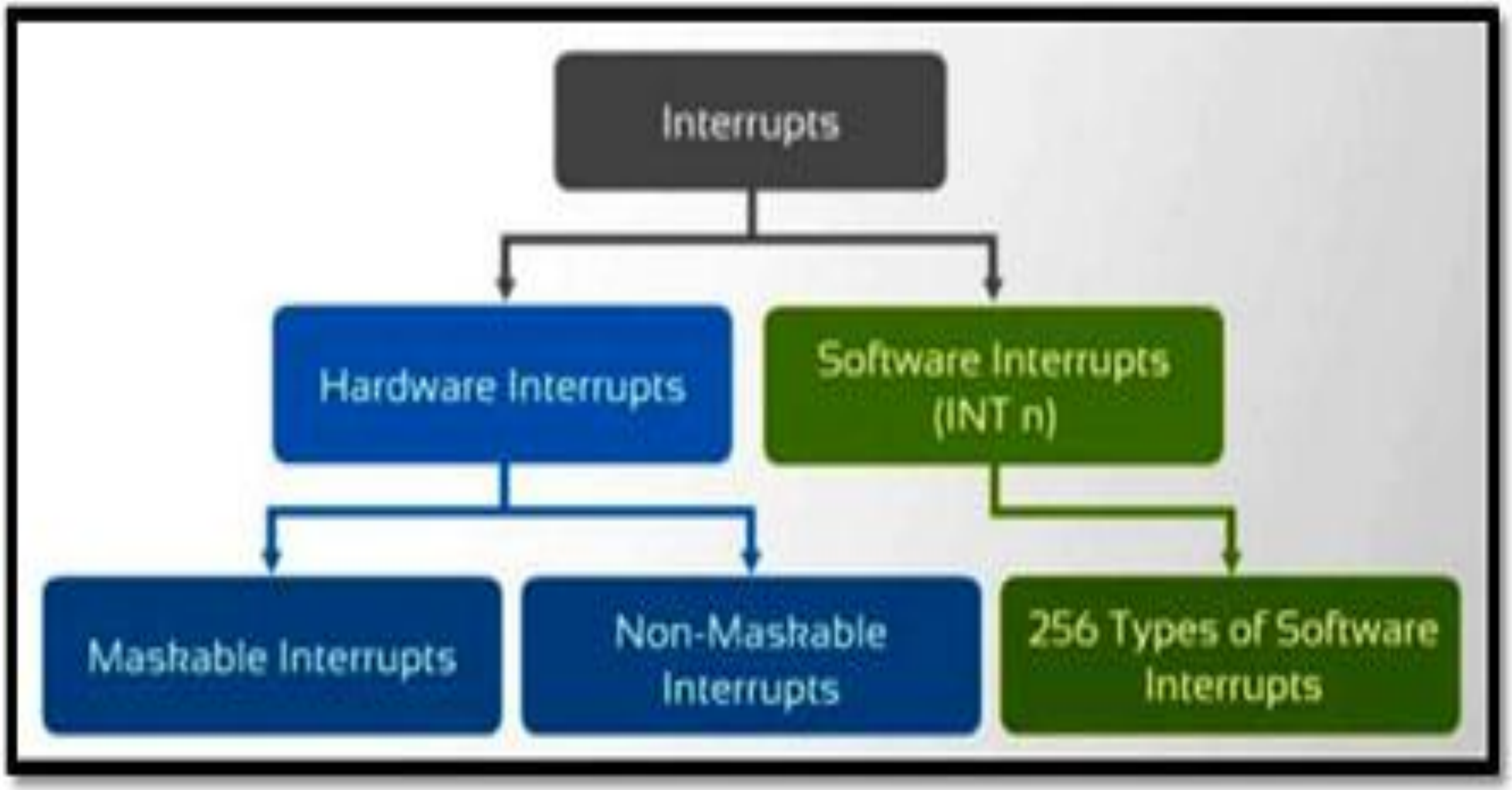
- The processor can be interrupted in the following ways:

  i) by an external signal generated by a peripheral, such as Keyboard, Mouse, Printer, etc.

  ii) by an internal signal generated by a special instruction in the program,

  iii) by an internal signal generated due to an exceptional condition which occurs while executing an instruction.

Classification of Interrupts



Hardware Interrupt

Hardware Interrupts:

- The interrupts initiated by external hardware by sending an appropriate signal to the interrupt pin of the processor is called hardware interrupt.
- The 8086 processor has two interrupt pins INTR (18) and NMI (17).
- The interrupts initiated by applying appropriate signal to these pins are called hardware interrupts of 8086.
- Hardware Interrupts Used to handle external hardware peripherals such as key boards , mouse , hard disks , floppy disks , DVD drivers, printers, mouse, DVD drivers, floppy disks, key boards etc.

Maskable and Non-Maskable

 The processor has the facility for accepting or rejecting hardware interrupts.

- **Maskable Interrupts** (Can be delayed or Rejected) The interrupts whose request can be either accepted or rejected or delayed by the processor are called maskable interrupts.
- Example: User-defined interrupts. In 8086 processor all the hardware interrupts initiated through INTR pin are maskable by clearing interrupt flag (IF). IF = 0, wait (CPU does not service the interrupt yet).

- **Non-Maskable Interrupts** (Can not be delayed or Rejected) The interrupts whose request has to be definitely accepted (or cannot be rejected) by the processor are called non-maskable interrupts. (Value of IF flag is IGNORED)
- Example: The interrupt initiated through NMI pin and all software interrupts are non-maskable.

Let say IF = 1

INTR = 1

Examples

- Non-Maskable Interrupts
  - Used during power failure
  - Used during critical response time
  - Used during non-recoverable hardware errors
  - Used during memory parity errors

- Maskable Interrupts
  - Interrupt from mouse, keyboard or other peripherals

Software Interrupts

- The software interrupts are program instructions.
- These instructions are inserted at desired locations in a program.
- While running a program, if software interrupt instruction is encountered then the processor initiates an interrupt.
- The 8086 processor has 256 types (00 to FF) or (00 to 255) of software interrupts. The software interrupt instruction is INT n, where n is the type number in the range 0 to 255.

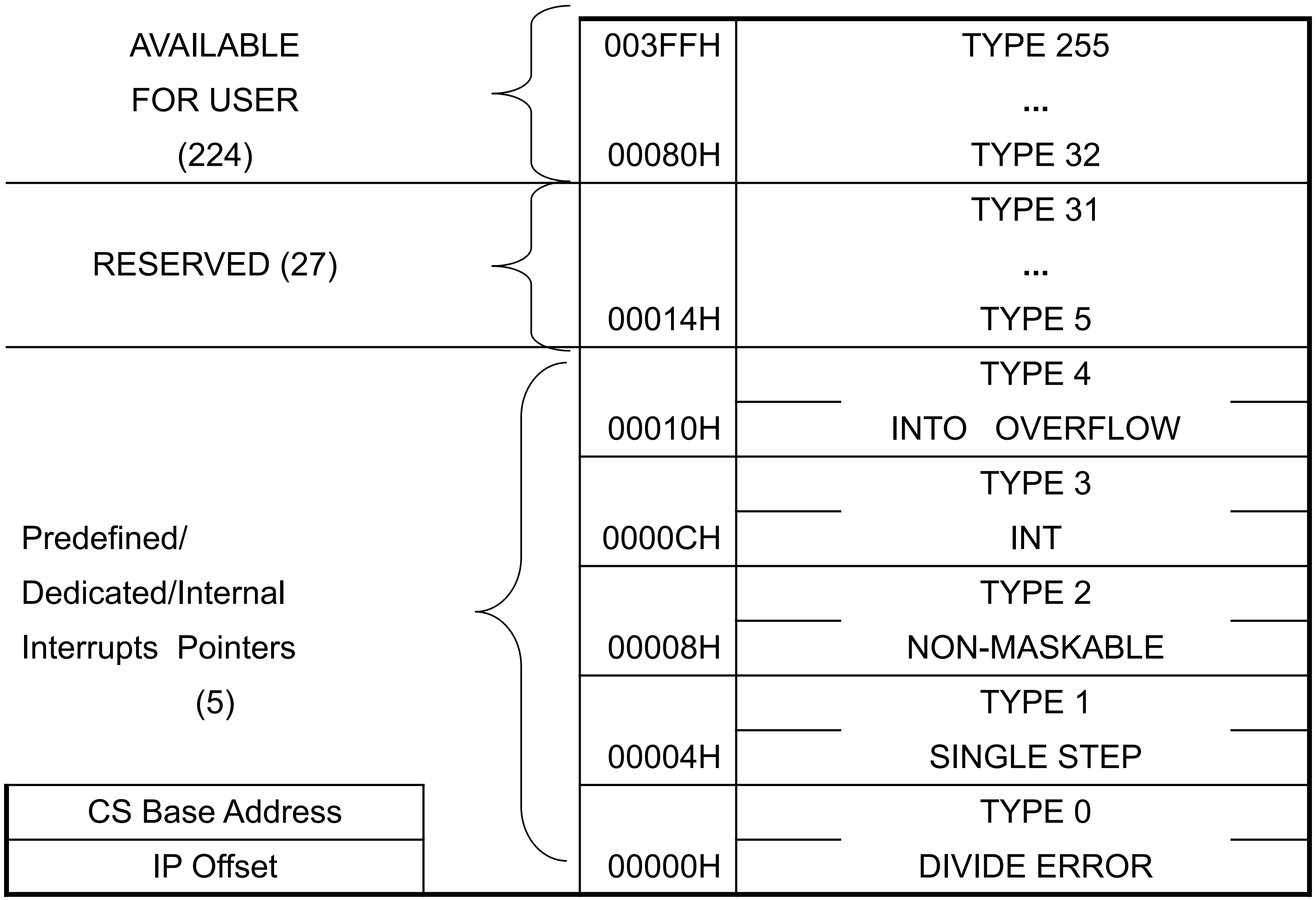
Classifications of 8086 Interrupts

- An 8086 interrupt can come from any of the ***three*** sources:
  - An ***external signal*** applied to NMI or INTR pin.
    - known as ***hardware interruption***
    - *It is a **user-defined interrupt***
  - Execution of interrupt instruction ***INT.***
    - referred as ***software interruption***
    - *It is also a **user-defined interrupt***
  - Some error condition produced by execution of an instruction, e.g., trying to divide some number by zero.
    - It is known as ***pre-defined interrupt***

Interrupt Vectors and Vector Table

- All interrupts (vectored or otherwise) are mapped onto a memory area called the **Interrupt Vector Table (IVT)**.
    - The IVT is usually located in the first 1 Kbyte of memory segment (from 00000 H - 003FF H).
    - The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.
- An **interrupt vector** is a **pointer** to where the ISR is stored in memory.
- The starting address of an ISR is often called
    - the ***interrupt vector*** or the ***interrupt pointer.***
- So the Table is referred to as
    - ***interrupt-vector table*** or ***interrupt-pointer table.***

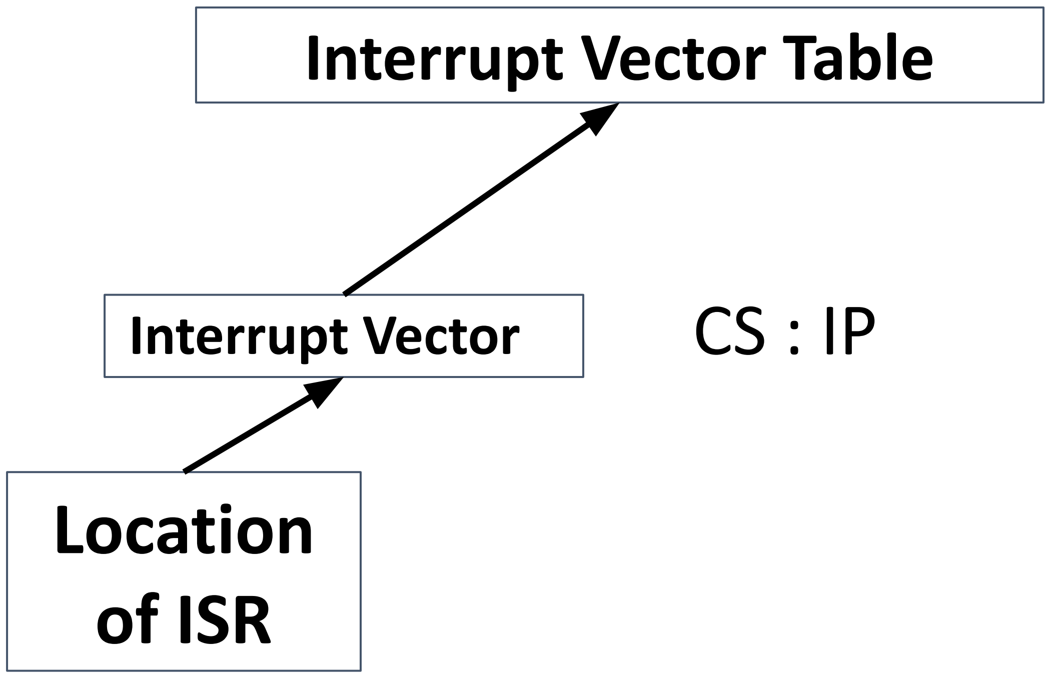
Interrupt Types based on ISR ID



AVAILABLE FOR USER (224)	003FFH	TYPE 255
		...
	00080H	TYPE 32
RESERVED (27)		TYPE 31
		...
	00014H	TYPE 5
Predefined/ Dedicated/Internal Interrupts Pointers (5)	00010H	TYPE 4 INTO OVERFLOW
	0000CH	TYPE 3 INT
	00008H	TYPE 2 NON-MASKABLE
	00004H	TYPE 1 SINGLE STEP
	00000H	TYPE 0 DIVIDE ERROR

CS Base Address
IP Offset

Overview Of ISR and IVT



CS : IP

1 address / location = 1 byte of data.

Address	00003h	00004h	00005h	00006h	00007h
Data	54h	12h	78h	34h	89h

Suppose, I am at pointing at 00004h and a store 2 bytes of data, I am now pointing at 00004h + 2h = 00006h.
Notice how it is 2h and not 2d (Decimal)

Example Of finding address of instruction

A (5 byte)	B (7 byte)	C (3 byte)

Starting location → 00013h

Location of A = 00013h

Location of B: 00013h + 0005h = 00018h

Location of C : Location of B + 0007h = 0001Fh

5 + 7 = 12d

Location of C : 00013h + 0012d = 25 (wrong answer)

Right Approach:

12d = 0Ch

Location of C : 00013 + 000Ch = 0001Fh

Interrupt Types based on ISR ID

- Note that
  - The **IP** value is put in as the **low word** of the vector
  - **CS** as **high word** of the vector
- 4 bytes are required to store the CS and IP values for each interrupt service procedure, the ***interrupt-vector table*** can hold starting addresses for up to 256 interrupt procedures.
- Each ***Double Word*** interrupt vector is identified by a number from 0 to 255
- *INTEL* calls this number the ***TYPE*** of the interrupt
- CS = 2 bytes, IP = 2 Bytes, so total 4 bytes are needed for Interrupt and there are total 256 interrupts, where 256 x 4 = 1024 Bytes = 1 KB are located in the starting part of memory
- CS:IP = 0000:0000 to 0000:03FF

How many interrupt vector in IVT

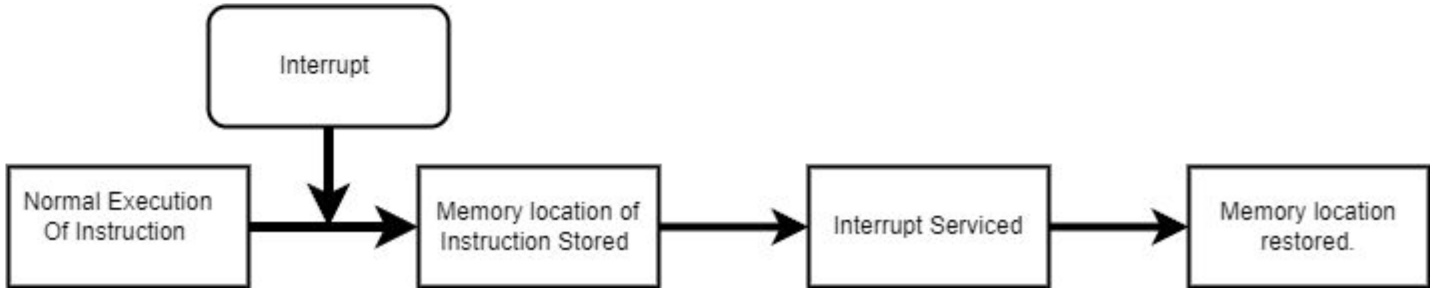
1024 bytes / 4 bytes = 256

Question

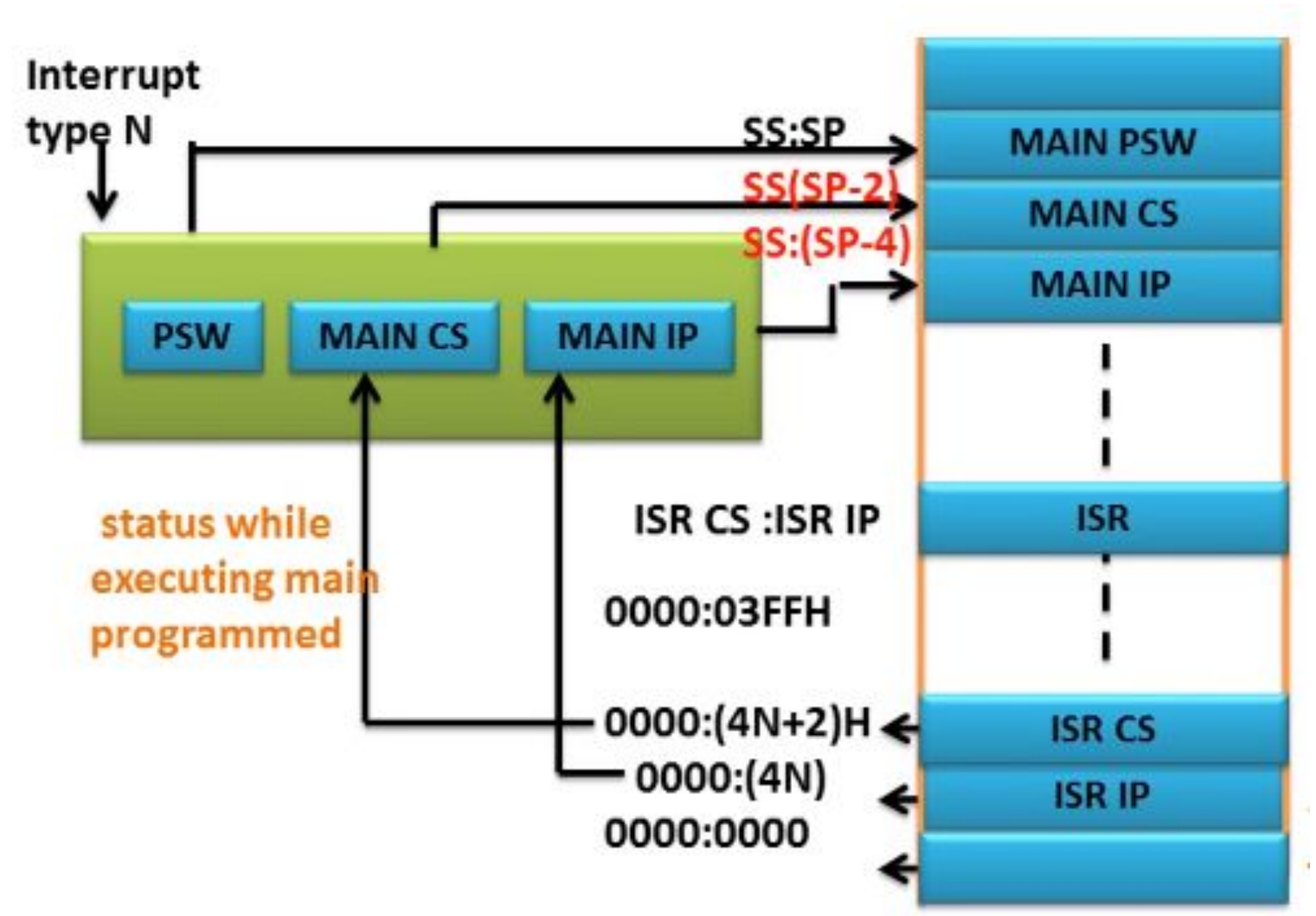
- Why is the interrupt vector of 4 byte?
- The IVT is 1KB. How many interrupt vector are there and why?
- What is stored inside the interrupt vector?

Normal Code with interrupt

1. Instruction 1
2. Instruction 2
3. Instruction 3
4. Interrupt 1
5. Instruction 4
6. Instruction 5


Interrupt
Normal Execution Of Instruction
Memory location of Instruction Stored
Interrupt Serviced
Memory location restored.

How non-maskable Interrupt is serviced



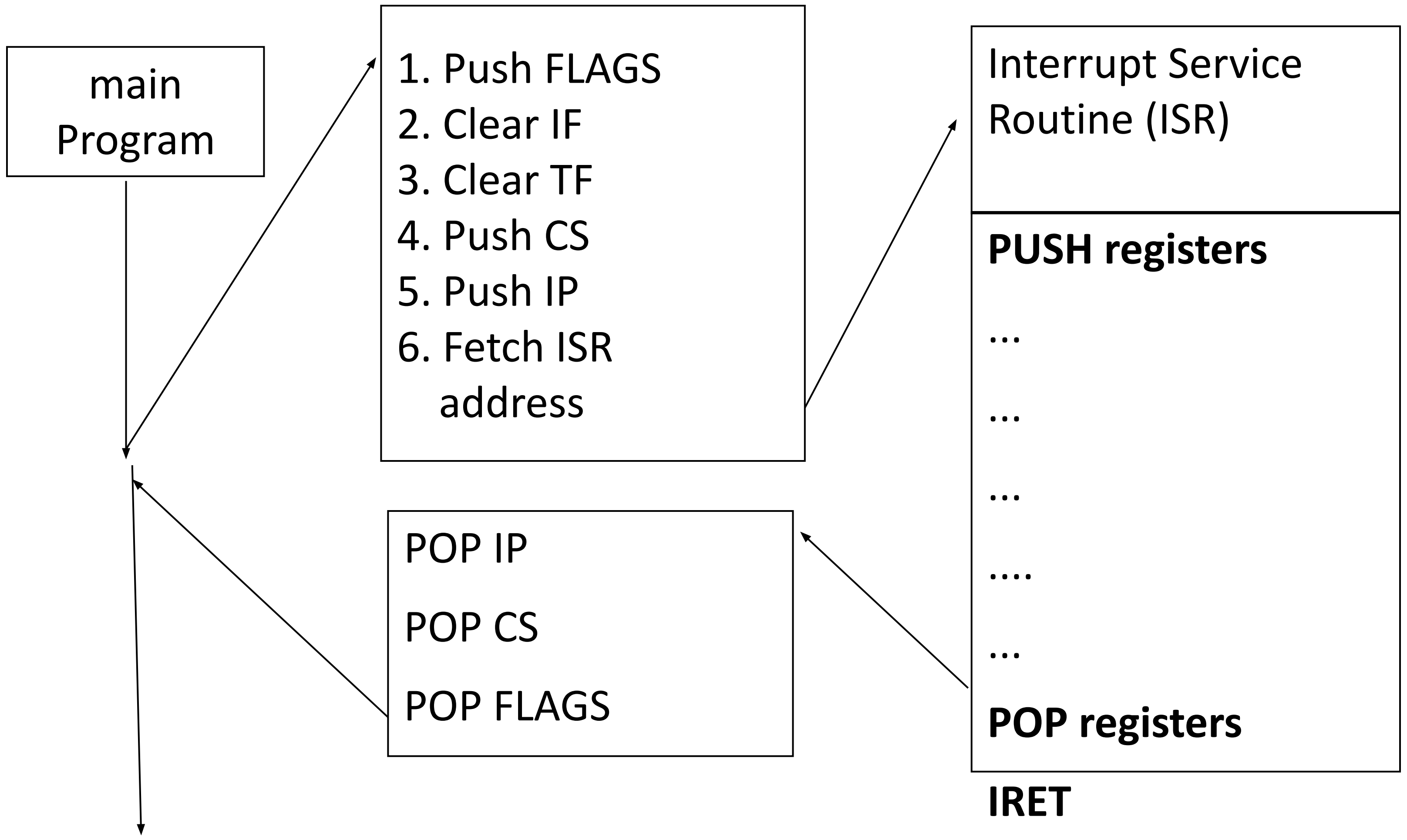
Function of 8086 during Interrupts

- At the end of each instruction cycle, 8086 checks to see if any interrupts have been requested.
- If yes, then 8086 responds to the interrupt by stepping through the following series of major actions:
  - It decremented SP by 2 and pushes ***Flag register*** on the stack.
  - It disables 8086 **INTR** input by clearing **IF (Interrupt) flag** in Flag register
  - It resets the **TF (Trap) flag** in Flag register
  - It decremented SP again by 2 and pushes current **CS (Code Segment)** contents on the stack.
  - It decremented SP again by 2 and pushes current **IP (Instruction Pointer)** contents on the stack.
  - It does an indirect far **Jump** to the start of the procedure written to respond to the interrupt.

Question

- Write down the full process about how the an interrupt is serviced (ans in last slide)
- Why is TF and IF cleared.

Function of 8086 during Interrupts



Calculation

- Now we know that the interrupt number (nn) is 60. So we can easily get the effective address from the interrupt vector table. For any interrupt number there are 4 line address first two for IP and last two for CS.
- The equation to get effective address from IP:CS is,

  IP = (nn x 4) and CS = (nn x 4) + 2
- So for 60, the RAM location of IP is (60 x 4) = 240 or 000F0h & 241 or 000F1h. And the RAM location of CS are 242 or 000F2h & 243 or 000F3h. The value of IP & CS are 012h & 0E8h respectively.
- Now from CS:IP we get the physical address where the interrupt service routine exist. Here the 20 bit physical address is 0E92h. Through this address we get the ISR and it will execute until the IRET which declare the last instruction of ISR.

Calculation

000EDh	000EEh	000EFh	000F0h	000F1h	000F2h	000F3h	000F4h	000F5h	000F6h
34h	24h	23h	**12h**	**00h**	**E8h**	**00h**	76h	65h	32h

IP

CS

IP - 0012h

CS - 00E8h

Location of our target ISR = 00E80 + 0012 = 00E92h

Finding INT type from CS value

000ECh	000EDh	000EEh	000EFh	000F0h	000F1h	000F2h	000F3h	000F4h	000F5h	000F6h
21h	34h	24h	23h	12h	00h	E8h	00h	76h	65h	32h

The CS we got is 6576h. Find the INT type?

6576h value is stored is in 00F4h and 00F5h.
So 00F4h and 00F5 is where CS values are stored.

IP values are in = 00F2h and 00F3h
So 00F2h in decimal is 242d.
242d / 4 = 60.5 So we have INT 60

Finding INT type from IP value

000ECh	000EDh	000EEh	000EFh	000F0h	000F1h	000F2h	000F3h	000F4h	000F5h	000F6h
21h	34h	24h	23h	12h	00h	E8h	00h	76h	65h	32h

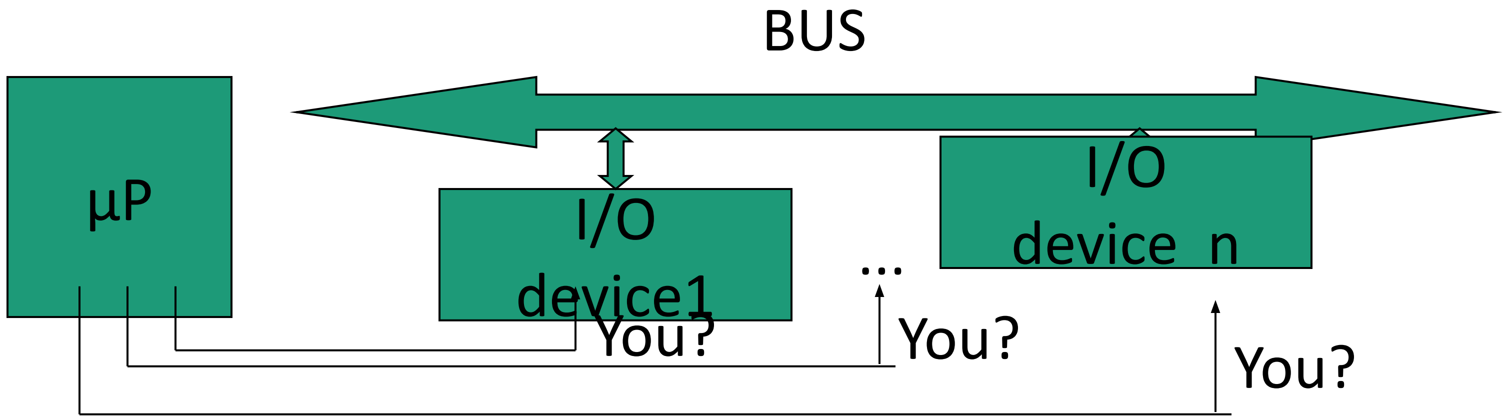
The IP value we got is 2324h. Find the INT type?

2324h value is stored is in 000EEh and 00EFh.
So 00EEh and 00EF is where IP values are stored

So 00EEh in decimal is 238d.
238 / 4 = 59.5 So we have INT 59

Why Interrupt is Necessary?

- **First Type:** POLLED I/O or PROGRAMMED I/O communication between MP and I/O devices.

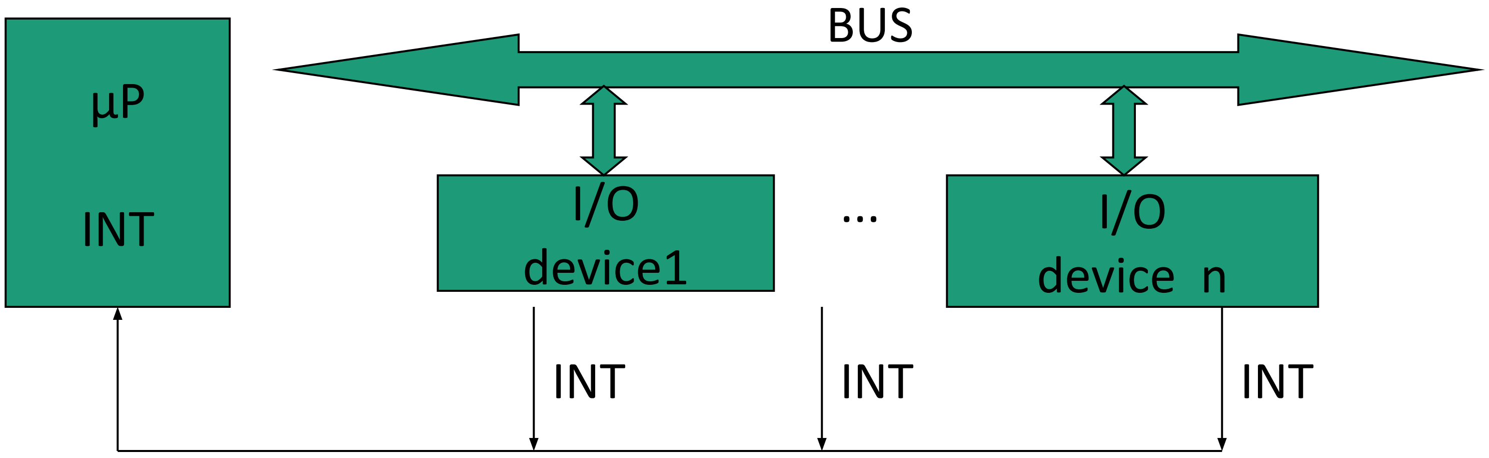


Disadvantages of Second Type Communication:

- not fast enough
- waste too much microprocessor time

Why Interrupt is Necessary?

- **Third Type:** INTERRUPTED I/O communication between MP and I/O devices.

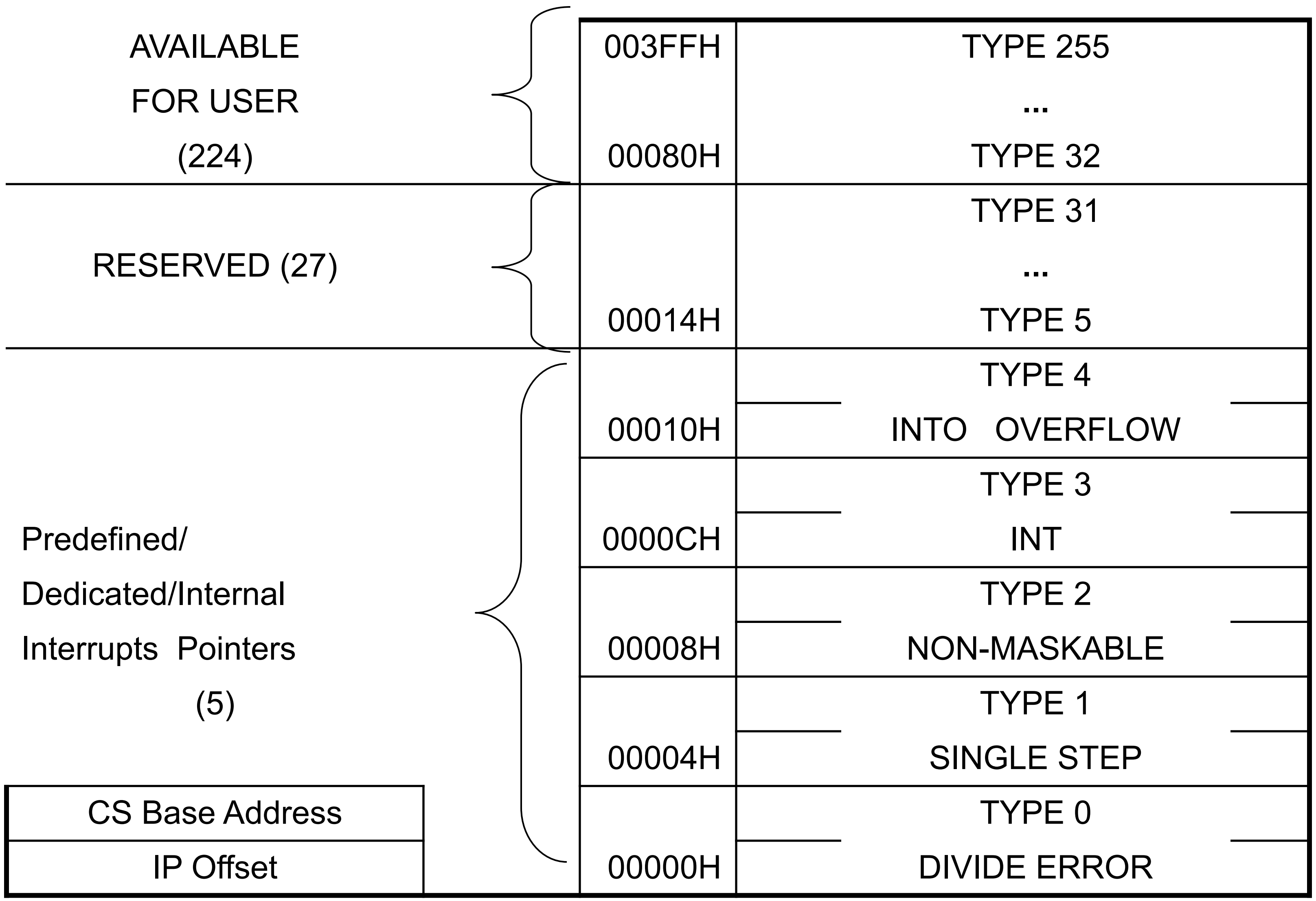


Interrupts are particularly useful when I/O devices are slow

Polling and Interrupt

- Both are methods to notify processor that I/O device needs attention
- **Polling**
  - simple, but slow
  - processor check status of I/O device regularly to see if it needs attention
  - *similar to checking a telephone without bells*!
  - Handled by CPU
- **Interrupt**
  - fast, but more complicated
  - processor is notified by I/O device (interrupted) when device needs attention
  - Handled by Interrupt - handler
  - *similar to a telephone with bells*

Interrupt Types based on ISR ID



AVAILABLE FOR USER (224)	003FFH	TYPE 255
		...
	00080H	TYPE 32
RESERVED (27)		TYPE 31
		...
	00014H	TYPE 5
Predefined/ Dedicated/Internal Interrupts Pointers (5)		TYPE 4
	00010H	INTO OVERFLOW
		TYPE 3
	0000CH	INT
		TYPE 2
	00008H	NON-MASKABLE
		TYPE 1
	00004H	SINGLE STEP
		TYPE 0
	00000H	DIVIDE ERROR

CS Base Address
IP Offset

Divide by zero interrupt- Type 0

- It occurs automatically when the result of DIV or IDIV is too large
- Example:

```
DIV BL              //This will do AX÷BL
        //will put result in AL(quotient) and AH(reminder)
```

If AX was 4000H and BL was 02H,

The quotient is 2000H and reminder is 00H.

µP can put 00H in AH(reminder)

But it cannot put 2000H in AL(quotient)

This condition is called divide error!

Again by mistake, we do a division operation where divisor is 0, such as AX = 4000H and BL = 00H, then the result will be infinity and it is the largest number that can not be stored in AL or AX whatever.

This is also divide by 0 error or Type 0 interrupt.

Single Step Interrupt- Type 1

- In single step mode, a system will **stop after it executes each instructions** and wait for further direction
- If the 8086 **trap flag** is set, the 8086 will automatically do a type 1 interrupt after each instruction executes
- The trap flag is reset when the 8086 does a type 1 interrupt, so the single step mode will be disabled during the interrupt-service procedure
- Tasks for implementing single stepping:
  - Set the trap flag
  - Write an interrupt service procedure which saves all registers on the stack
  - The stack where they can later be examined
  - Load the starting address of the type 1 interrupt service procedure into address 00004H and 00006H

Non-Maskable Interrupt- Type 2

- The 8086 will automatically do a type 2 interrupt response when it receives a low to high transition on its NMI input pin
- The 8086 gets the CS value for the start of the type 2 interrupt service procedure from address 0000AH and the IP value for the start of the procedure from address 00008H
- **The type 2 interrupt response cannot be disabled** by any program instruction that's why we use it to signal the 8086 that some condition in an external system must be taken care of
- **The type 2 interrupt is useful to save program data in case of a system power failure**

Breakpoint Interrupt- Type 3

- The main use of type 3 interrupt is **to implement a breakpoint function** in a system
- Unlike the single step feature, which stops execution after each instruction, the breakpoint feature **executes all the instruction up to the inserted breakpoint** and then stops execution
- The 8086 gets the CS value for the start of the type 3 interrupt service procedure from address 0000EH and the IP value for the start of the procedure from address 0000CH
- **It is useful in debugging large programs when single stepping is inefficient**

Overflow Interrupt- Type 4

- The 8086 **overflow flag (OF) will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented or memory location**
- The 8086 will get the CS value for the start of the type 4 interrupt service procedure from address 00012H and the IP value for the start of the procedure from address 00010H
- It is **useful to detect overflow error** in signed arithmetic operations
- There are **two ways to detect and respond** to an overflow error in a program
  - **Put the jump if overflow instruction (JO)** immediately after the arithmetic instruction
  - **Put the interrupt on overflow instruction (INTO)** immediately after the arithmetic instruction

Summary of 8086 Interrupt Function …

- **What happens if two or more interrupts occur at the same time?**
  - Higher priority interrupts will be served first

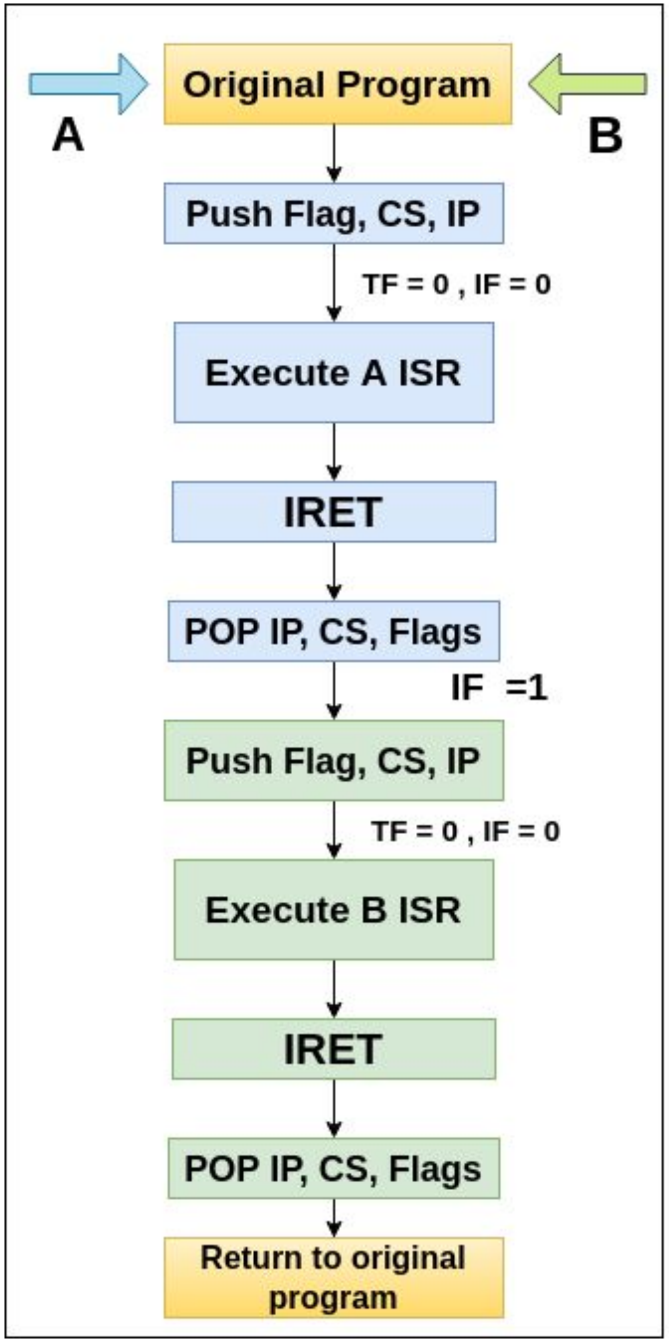
Interrupt Type	Priority
DIVIDE ERROR, INTn, INT0	HIGHEST
NMI (does not wait)	
INTR	
SINGLE STEP	LOWEST

Scenarios that can happen

1. A single interrupt
2. Multiple interrupt coming at once / simultaneously
3. One interrupt is being serviced, lower priority interrupt occurs
4. One interrupt is being serviced, higher priority interrupt occurs

Flow Chart: When two interrupts with different priority occur at the same time

Priority: A > B



Flow Chart: When one interrupt is being serviced and another low priority interrupt occurs

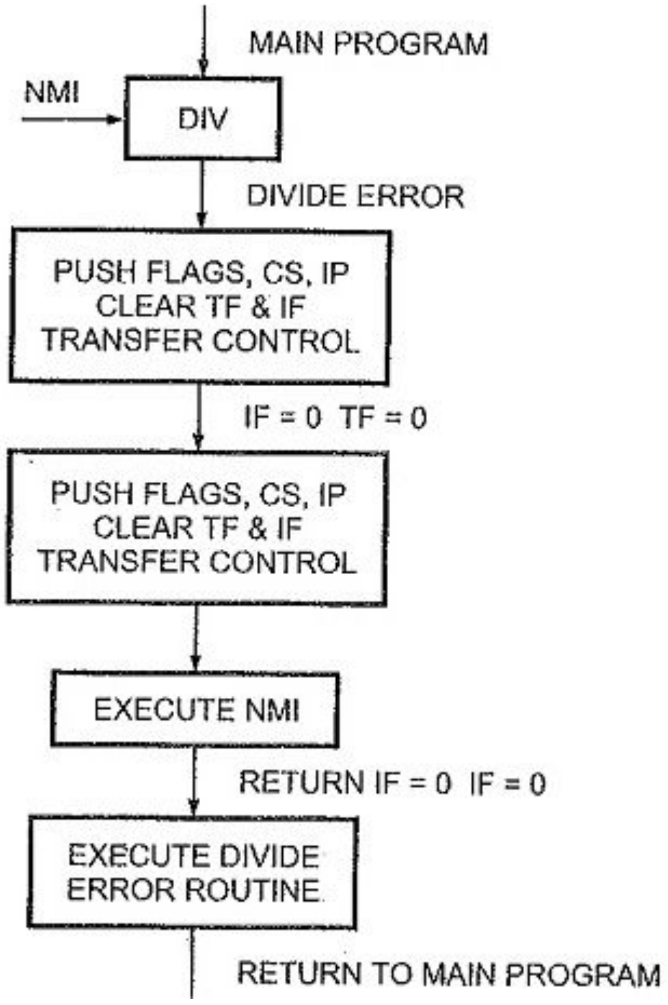
Priority: A > B

Same as the previous slide. However B will come after A. Not with A. So the arrow of B will be after the first box.

Flow Chart: When one higher priority interrupts comes during execution of another interrupt

DIV →

NMI →


MAIN PROGRAM
NMI
DIV
DIVIDE ERROR
PUSH FLAGS, CS, IP CLEAR TF & IF TRANSFER CONTROL
IF = 0 TF = 0
PUSH FLAGS, CS, IP CLEAR TF & IF TRANSFER CONTROL
EXECUTE NMI
RETURN IF = 0 IF = 0
EXECUTE DIVIDE ERROR ROUTINE
RETURN TO MAIN PROGRAM

Question

- Can two different interrupt be serviced at a time.
- If NMI & INTR interrupt happen simultaneously, which interrupt is serviced first.
- IF INTR interrupt occur first and then NMI, which interrupt is serviced first.
- IF INTR and Single Step interrupt simultaneously. And then NMI interrupt occurs, write the order of interrupt serviced.
- If Single step and then INTR interrupt happens. And then simultaneously Divide Error and NMI interrupt happens. Write the order of interrupt being serviced.